ISSN 2186-5647

5-56

並列処理計算を用いたナッシュ均衡点計算の効率化へ向けて

日大生産工(院)　○高田　貴文　日大生産工　吉田　典正

1. 目的

本研究では, ゲーム理論における非協力ゲームの解の一種であるナッシュ均衡点を並列処理によって計算し,効率化の度合いを測定することを目的とする.具体的には,文献[1]の混合戦略のナッシュ均衡点の計算時間を短縮するために, CPUの複数のコアを利用した並列処理計算の手法を用いたナッシュ均衡点を求めるプログラムを作成し, 戦略数を(2×2)～(15×15)行列に変化させ, 計算時間の測定を行い, 並列処理の効果を測定し, どれ程短縮されたのかを測る.

2. ナッシュ均衡点の計算

ナッシュ均衡点の計算方法には純粋戦略と混合戦略の2種類があり, ここでは, 表1の利得行列が与えられた際の混合戦略の計算方法を説明する. なお, 表の左の値はプレイヤーAの利得, 右の値はプレイヤーBの利得である.

表1　2×3の利得行列

A / B	B1	B2	B3
A1	1　−3	−2　6	8　7
A2	2　1	3　0	−3　−2

プレイヤーAとプレイヤーBの戦略の集合をそれぞれ $M=\{0,\ldots,m-1\}, N=\{m,\ldots,m+n-1\}$ とし, M,N の k 個ずつの要素を持つ部分集合を I,J とする. k とはサポートサイズと呼ばれ, 互いの戦略の中からいくつの戦略を選択するかを表し, 混合戦略では $k=2,\ldots,\min(m,n)$ である. 純粋戦略では $k=1$ である. ここで, k 個ずつの戦略を持つ部分集合 I,J が与えられたとする. 表1の行列では, プレイヤーAとプレイヤーBの戦略の集合 M,N はそれぞれ $M=\{0,1\}, N=\{2,3,4\}$ となっており, 混合戦略におけるサポートサイズは $k=2$ だけであるので, 部分集合 I,J の組み合わせは次の3通りである.

$$\left(I=\{0,1\},J=\{2,3\}\right),\left(I=\{0,1\},J=\{3,4\}\right),\left(I=\{0,1\},J=\{2,4\}\right) \quad (1)$$

次に, プレイヤーAとプレイヤーBの混合戦略を求める. プレイヤーAの混合戦略を x, プレイヤーBの混合戦略を y とすると,

$$x=(x_0,\ldots,x_{m-1}),\ y=(y_m,\ldots,y_{m+n-1}) \quad (2)$$

となる. (1)に示したすべての組み合わせに対して, 式(3)を満たすような x,y を計算し, プレイヤーAの混合戦略 x と, プレイヤーBの混合戦略 y を求める.

$$\sum_{i\in I}x_i b_{ij}=v \text{ for } j\in J,\ \sum_{i\in I}x_i=1$$
$$\sum_{j\in J}a_{ij}y_j=u \text{ for } i\in I,\ \sum_{j\in J}y_j=1 \quad (3)$$

$I=\{0,1\},\ J=\{2,3\}$ の場合の混合戦略の計算方法を式(4), (5)に示す.

$$\begin{bmatrix}x_0 & x_1 & v\end{bmatrix}\begin{bmatrix}-3 & 6 & 1\\ 1 & 0 & 1\\ -1 & -1 & 0\end{bmatrix}=\begin{bmatrix}0 & 0 & 1\end{bmatrix} \quad (4)$$
$$\Rightarrow x_0=0.1,\ \ x_1=0.9,\ \ v=0.6$$

$$\begin{bmatrix}1 & -2 & -1\\ 2 & 3 & -1\\ 1 & 1 & 0\end{bmatrix}\begin{bmatrix}y_2\\ y_3\\ u\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 1\end{bmatrix} \quad (5)$$
$$\Rightarrow y_2=1.25,\ \ y_3=-0.25,\ \ u=1.75$$

混合戦略は確率分布に基づくものであるから, それぞれの混合戦略 x,y は

$$0\le x_i\le 1,\quad 0\le y_j\le 1 \quad (6)$$

を満たさなければならない.従って,式(4), (5)で求めた点はナッシュ均衡点にならない.

x,y のすべての要素が0から1までの範囲にあれば, 最適応答戦略になっているかどうかを調べる. x が y に対して最適応答戦略であるとは式(7)の条件式を満たすことである. y についても同様である.

Towards Efficient Computation of Nash Equilibrium using Parallel Processing

Takafumi TAKADA, Norimasa YOSHIDA

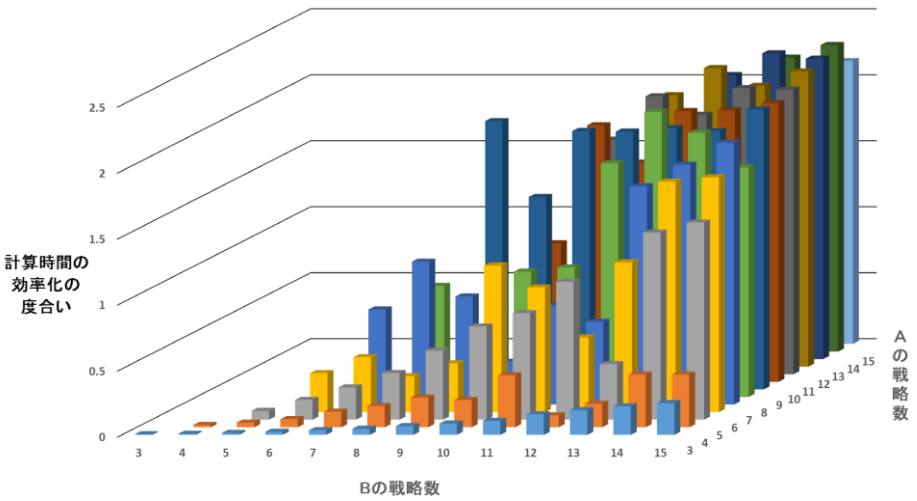

図1　マルチスレッド･プログラムの計算時間の効率化の度合い

$$x_i > 0 \Rightarrow (Ay)_i = u = \max\{(Ay)_k \mid k \in M\} \qquad (7)$$

I, J のすべての組み合わせの中で式(6)および最適応答戦略の条件式を満たしているのは $I = \{0,1\}, J = \{3,4\}$ の場合なので，表1の利得行列の混合戦略におけるナッシュ均衡点は $\left(\left(\frac{2}{3},\frac{1}{3}\right),\left(0,\frac{11}{16},\frac{5}{16}\right)\right)$ である．

3.　計算結果

本研究では，混合戦略を求めるプログラムをマルチスレッド･プログラムとして実装し，複数のコアを持つCPU上で動作させることで計算時間の短縮を図る．スレッドとはプログラムの具体的な処理のことであり，マルチスレッド･プログラムとは，複数の処理を同時に実行するプログラムのことである．処理を複数のスレッドに分け，各コアで並列処理させることによって，計算が効率化し，戦略数が多い場合に計算時間が短縮されることが期待できる．文献[1]で使われたスレッドを利用しないプログラムと，今回作成したマルチスレッド･プログラムの計算結果の比（スレッドを使わない場合の計算時間／マルチスレッドによる計算時間）を図1に示す．実験では，プレイヤーA，Bの利得を乱数で与え，A，Bの戦略数が(2×2)～(15×15)行列の場合の計算時間を3回ずつ繰り返し測定し，それらの平均値から上記の比を求め，その比をグラフにした．図１のグラフは，縦軸が1より大きい場合に，より効率化できていることを示し，1より小さい場合に，より効率的できていないことを表している．この図から，基本的にはAの戦略数またはBの戦略数が増えるほど効率化の度合いが高いことが分かる．なお，用いたCPUはIntel(R) Core(TM) i7-3770K CPU @ 3.50GHzであり，コア数は4，スレッド数は4である．また(2×3)行列と(3×2)行列は等しいため，計算結果は省略している．戦略数がより少ない場合に，より効率化できていない理由は，マルチスレッドで処理する場合には，その処理を行う為の前処理を行う必要があり，その処理に時間がかかるためである．

4.　まとめ

本研究では $m \times n$ 行列のナッシュ均衡点の計算の効率化を目的として，並列処理計算を行うプログラムを作成し，そのプログラムの計算時間と並列処理計算を行わないプログラムの計算時間との比較を行った．この結果，戦略数が多い程，効率化できる結果となった．今後の展望としては，図1における(5×13)行列の計算結果の比などの不自然に短いグラフが生じる原因と(9×9)行列の計算結果の比などの不自然に飛び出ているグラフが生じる原因を探る．さらに，GPUを用いた並列処理計算を行うことで，さらなる効率化を行おうと考えている．

「参考文献」

1) 平山圭太，m×n行列のナッシュ均衡点の計算と利得行列が変化した場合の再計算，日本大学生産工学部，卒業論文，2011.

2) 武藤滋夫，ゲーム理論入門，日本経済新聞出版社，2001.

3) Bernhard von Stengel, Equilibrium Computation for Two-Player Games in Strategic and Extensive Form, in Algorithmic Game Theory, edited by N. Nisan, et al., Cambridge University Press, 2007.